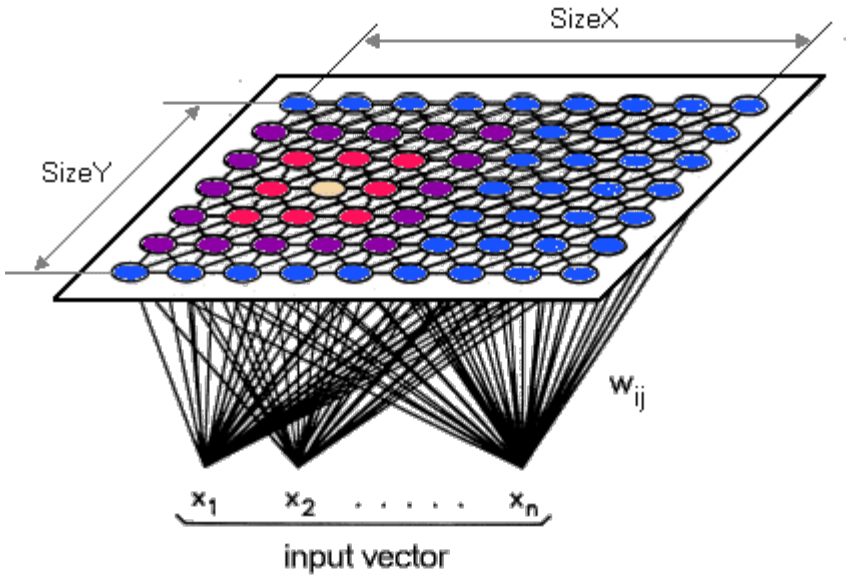
Unsupervised learning and Self Organizing Maps



Machine intelligence

Dr. Ahmad Al-Mahasneh

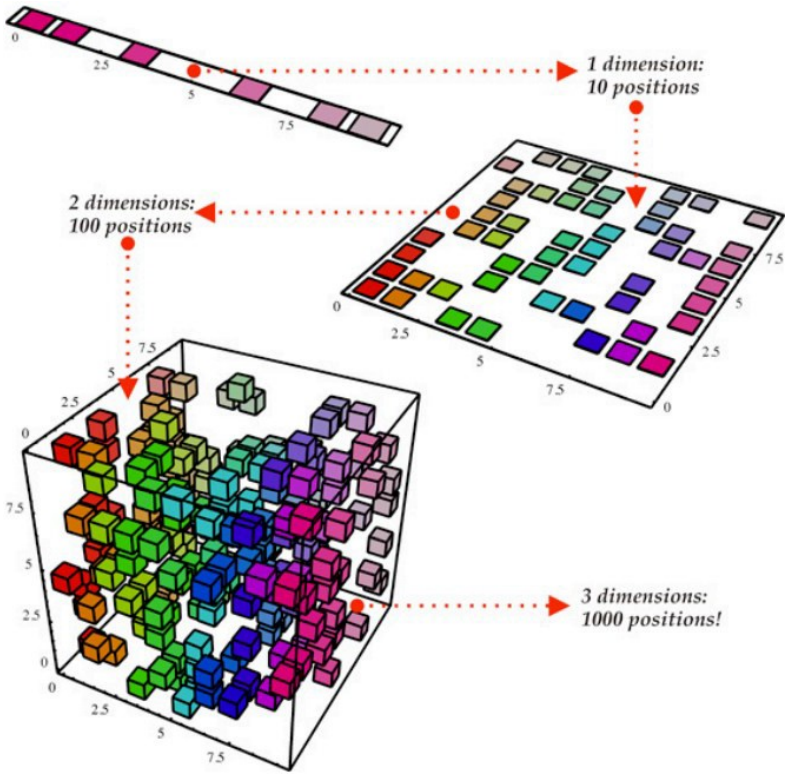
Outline

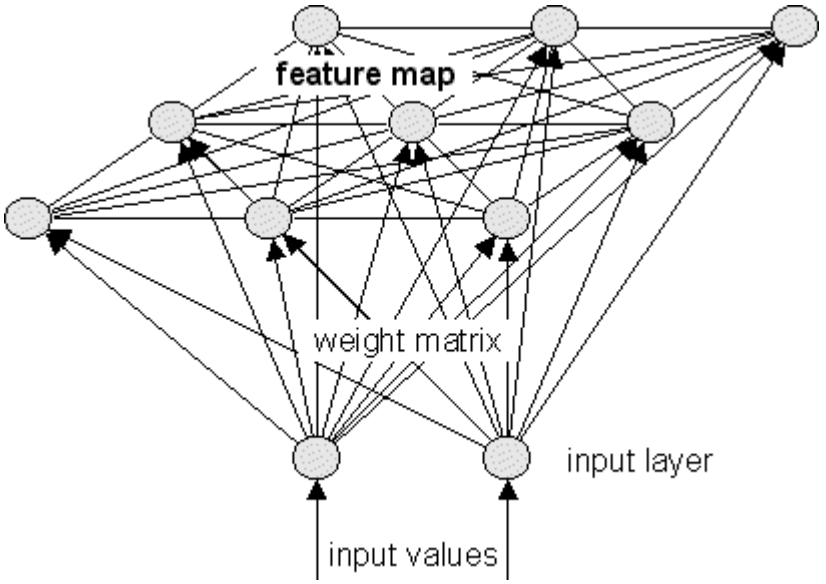
- Self-Organized Mapping
- Winner-takes-all SOMS
- Network Example
- Applications

Self-Organizing Map (SOM)

- SOM is to transform an incoming signal pattern of arbitrary dimension into one-or two-dimensional discrete map.
- SOM was originally invented by Kohonen in the mid 1990's
  - sometimes referred to as Kohonen Networks
- A SOM is a multi-dimensional scaling technique which constructs an approximation of the probability density function of some underlying data set and preserves the topological structure of that data set.
- This is done by mapping input vectors, in the data set , to weight vectors, neurons in the feature map.

Self-Organized Map (SOM)





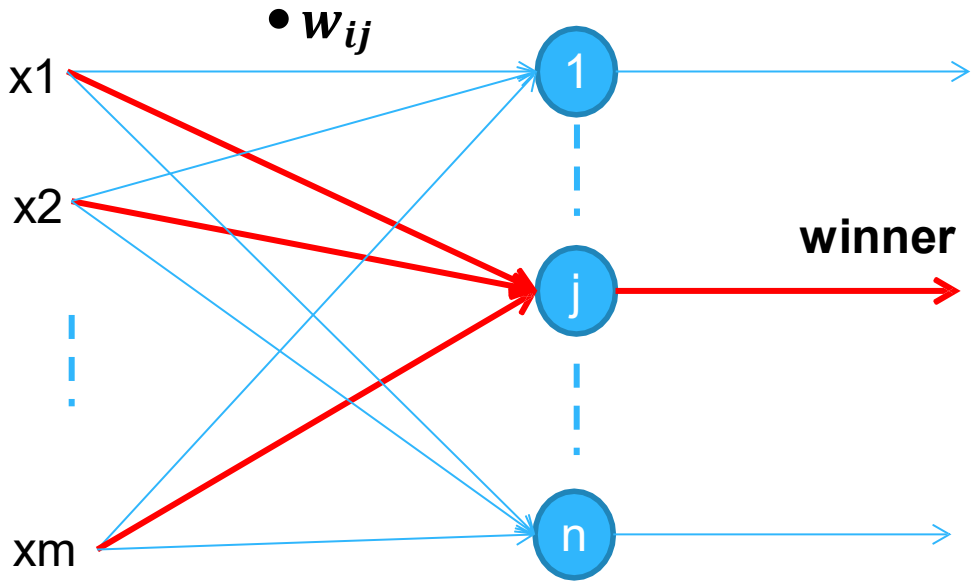
Source: Turingfinance.com

Competitive Learning

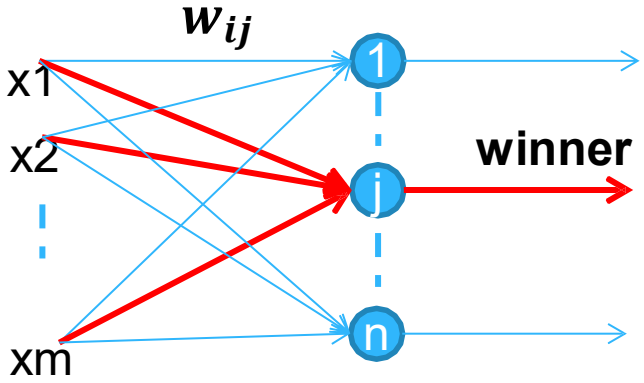
- A set of neurons are fully connected to the inputs
- Neurons compete for the best response to input patterns
- A mechanism for this competition should be decided
  - such as Euclidean distance or Mahalanobis distance
- Typically, a winner-takes-all method is used
  - The idea is one of the neurons will have the maximum response and therefore only its weights are adjusted

Winner-takes-all

- Winner-take-all algorithm works with single node in a layer of nodes that responds most strongly to the input pattern.



SOM Algorithm winner takes all



Initialize weights

Iteration Loop

Pattern Loop

Calculate Euclidean distance

$$D_j = \sum_{i=1}^{m} \left(x_i - w_{ij}\right)^2 \quad for\ j = 1{:}n$$

Determine the winner neuron

$$\boldsymbol{min\ D_j}$$

Adjust the weights for the winner neuron only

$$\boldsymbol{w_{ij} = w_{ij} + \alpha(x_i - w_{ij})}$$

End Pattern Loop

End Iteration Loop

SOM Algorithm with neighborhood function

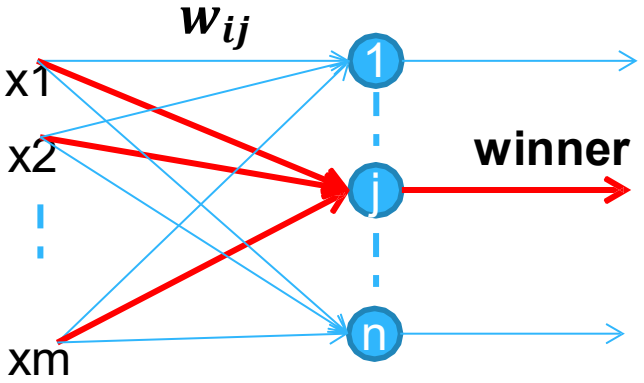
Initialize weights

Iteration Loop

Pattern Loop

Calculate Euclidean distance

$$D_j = \sum_{i=1}^{m} (x_i - w_{ij})^2 \quad for\ j = 1{:}n$$



Determine the winner neuron

$\boldsymbol{min\ D_j}$

Adjust the weights for the winner neuron only

$\boldsymbol{w_{ij} = w_{ij} + h_{ij}(x_i - w_{ij})}$

Where h_{ij} is the neighbourhood function

$$h_{ij} = \alpha(t)\mathrm{e}^{\left(\frac{-||r_i - c_i||}{2\sigma^2}\right)}$$

Where $0 < \alpha(t) < 1$ *is the learning rate parameter* σ is the width of the kernel

$$\alpha(t) = 0.9\left(1 - \frac{t}{1000}\right)$$

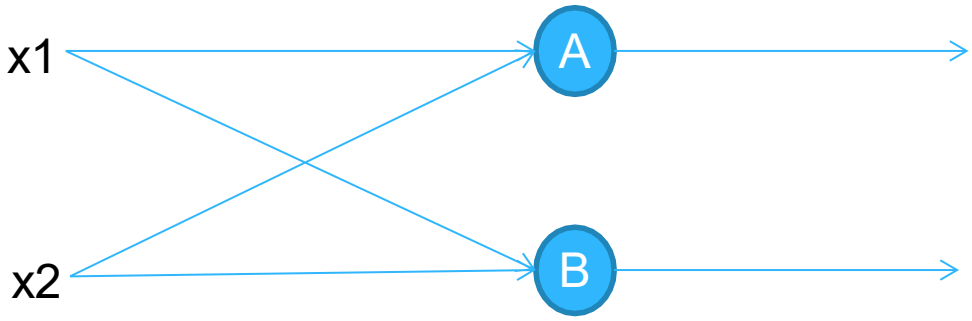
End Pattern Loop

End Iteration Loop

Example

- Consider the 2-dimensional data in table below
- Use two winner-takes-all neurons to classify the data as shown in figure below

X1	X2
1.0	1.0
9.4	6.4
2.5	2.1
8.0	7.7
0.5	2.2
7.9	8.4
7.0	7.0
2.8	0.8
1.2	3.0
7.8	6.1



Initialize the neuron weights as **WA** = [5 ; 3] and **WB** = [6 ; 8]

Example

Iteration One .. Pattern One

1. Competition:

Calculate the distance between the input pattern and neuron A

$$D_1{}^2 = (x_1 - w_{1A})^2 + (x_2 - w_{2A})^2$$

$$D_1{}^2 = (1 - 5)^2 + (1 - 3)^2 = 20$$

Calculate the distance between the input pattern and neuron B

$$D_2{}^2 = (x_1 - w_{1B})^2 + (x_2 - w_{2B})^2$$

$$D_2{}^2 = (1 - 6)^2 + (1 - 8)^2 = 74$$

Compare

$$D_1 < D_2 \Longrightarrow \boldsymbol{winner}\ is\ neuron\ \boldsymbol{A}$$

Example

2. Update Weights

Only weights of the winner neuron are updated

$$W_A = W_A + \alpha \ (X - W_A)$$

$$W_A = \begin{bmatrix} 5 \\ 3 \end{bmatrix} + 0.5 \left(\begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 5 \\ 3 \end{bmatrix} \right) = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

$$W_B = \begin{bmatrix} 6 \\ 8 \end{bmatrix}$$ unchanged

Example

Iteration One .. Pattern Two

1. Competition:

Calculate the distance between the input pattern and neuron A

$$D_1{}^2 = (x_1 - w_{1A})^2 + (x_2 - w_{2A})^2$$

$$D_1{}^2 = (9.4 - 3)^2 + (6.4 - 2)^2 = 112$$

Calculate the distance between the input pattern and neuron B

$$D_2{}^2 = (x_1 - w_{1B})^2 + (x_2 - w_{2B})^2$$

$$D_2{}^2 = (9.4 - 6)^2 + (6.4 - 8)^2 = 14$$

Compare

$$D_2 < D_1 \Longrightarrow \boldsymbol{winner}\ is\ neuron\ B$$

Example

2. Update Weights

Only weights of the winner neuron are updated

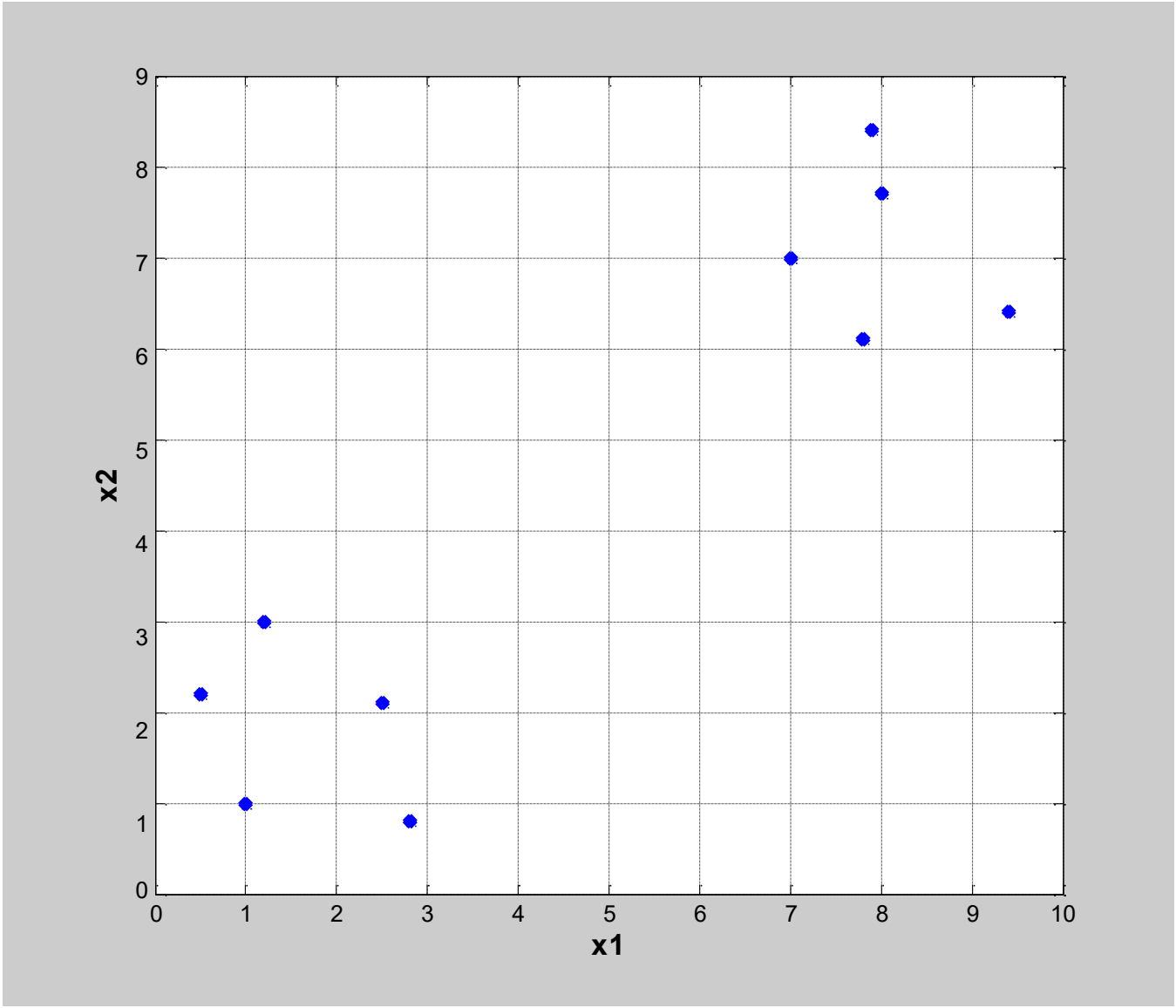
$$W_B = W_B + \alpha \ (X - W_B)$$

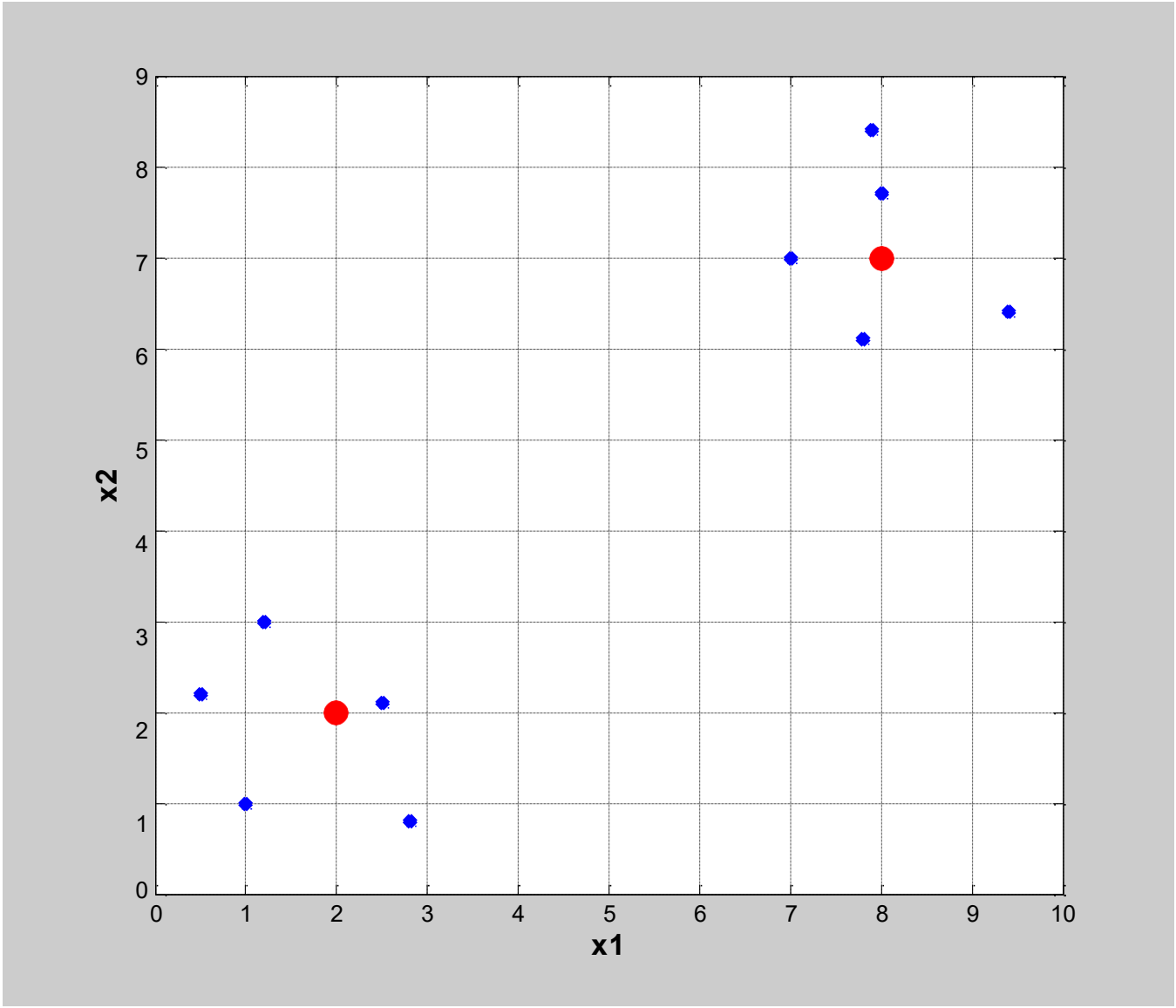
$$W_B = \begin{bmatrix} 6 \\ 8 \end{bmatrix} + 0.5 \left(\begin{bmatrix} 9.4 \\ 6.4 \end{bmatrix} - \begin{bmatrix} 6 \\ 8 \end{bmatrix} \right) = \begin{bmatrix} 7.7 \\ 7.2 \end{bmatrix}$$

$$W_A = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$ unchanged

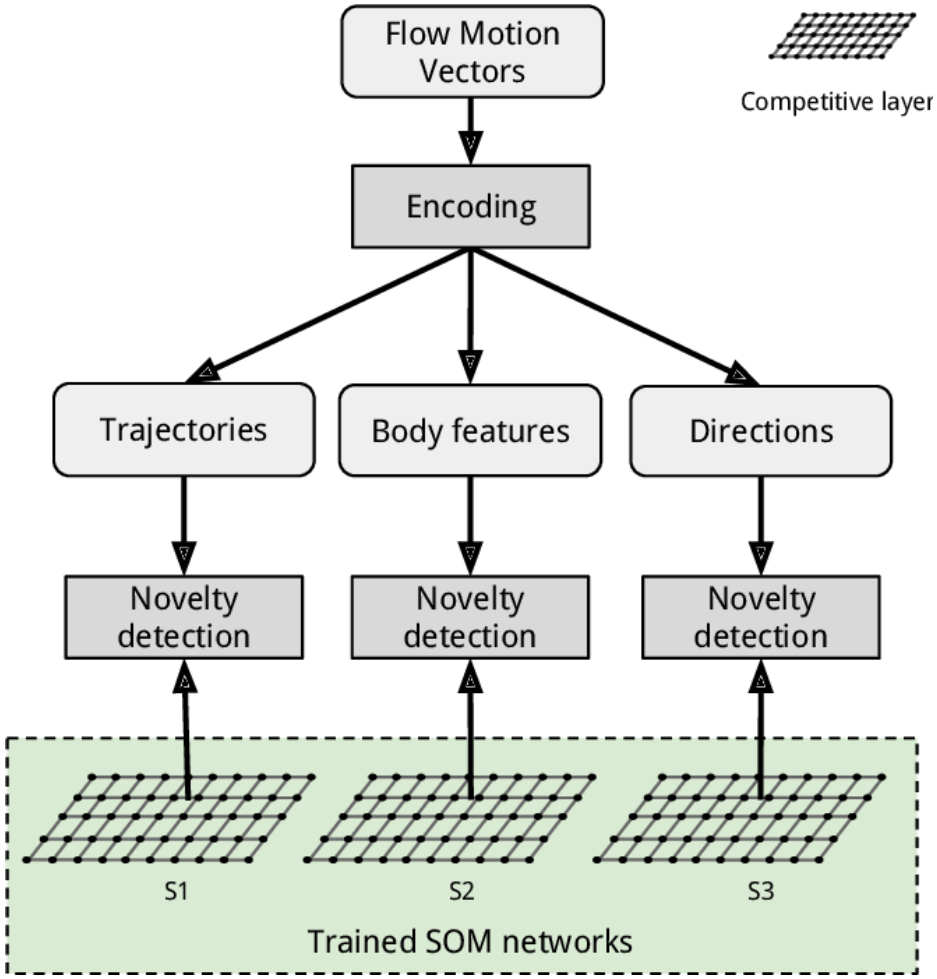
Example

- Iteration One
  - Pattern 3; Competition; Update
  - Pattern 4; Competition; Update
  - …
  - Pattern 10; Competition; Update
- Iteration Two
  - Pattern 1; Competition; Update
  - …
  - Pattern 10; Competition; Update
- Iteration Three
  - …

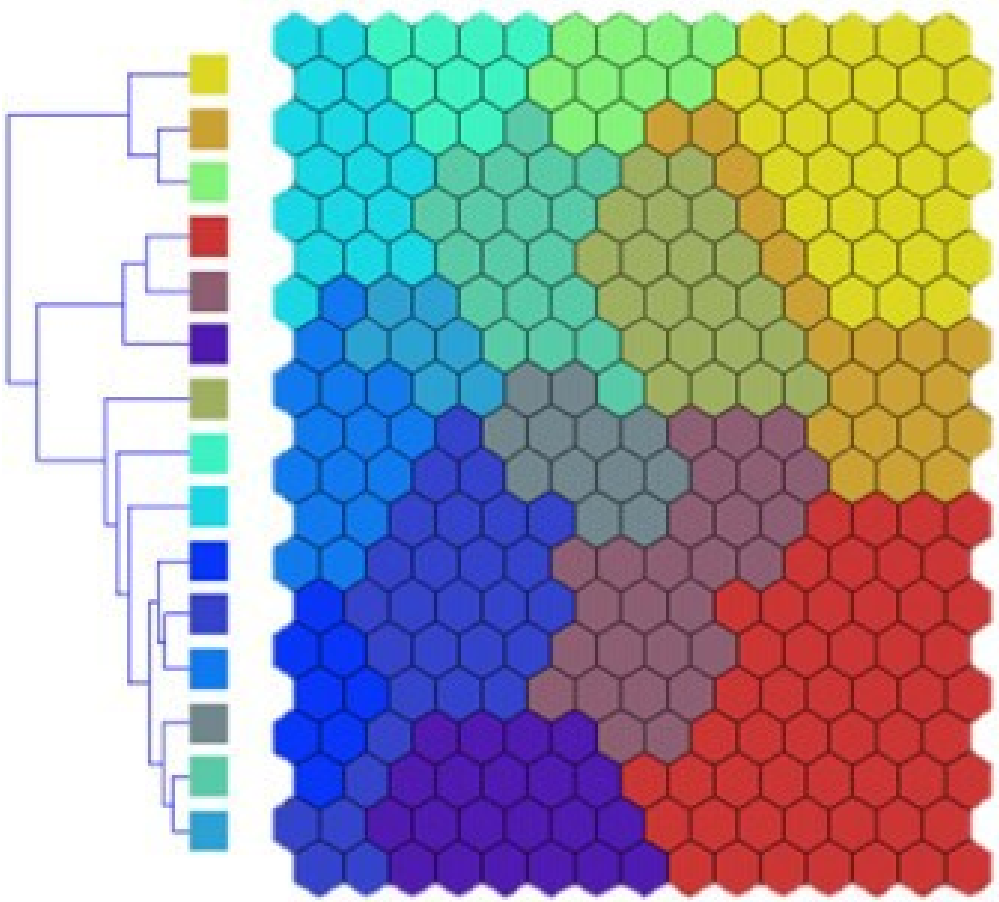

x2
x1
0
1
2
3
4
5
6
7
8
9
10


x2
x1
0
1
2
3
4
5
6
7
8
9
10

Application Example



Application Example



References

- Computational Intelligence: Synergies of Fuzzy Logic, Neural Networks and Evolutionary Computing (Chapter 4) by Siddique and Adeli. Wiley Publishing 2013

- Neural Networks and Learning Machine (Chapter 3) by Simon Haykin 3rd Edition. Pearson 2009